Guard Pages

Daniel Plakosh, Software Engineering Institute [vita[1]]

2005-09-27; Updated 2008-10-06 L2 / D/P, L[2]

Automatic allocation of additional inaccessible memory during memory allocation operations is a technique for mitigating against exploitation of heap buffer overflows. These guard pages are unmapped pages placed between all memory allocations of one page or larger. The guard page causes a segmentation fault upon any access.

Development Context

Dynamic memory management

Technology Context

C++, C, UNIX, Win32

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Standard C dynamic memory management functions such as `malloc()` , `calloc()`, realloc(), and `free()` [ISO/IEC 99] are prone to programmer mistakes that can lead to vulnerabilities resulting from buffer overflow in the heap, writing to already freed memory, and freeing the same memory multiple times (e.g., double-free vulnerabilities).

Description

Automatic allocation of additional inaccessible memory during memory allocation operations is a technique for mitigating against exploitation of heap buffer overflows. These guard pages are unmapped pages placed between all allocations of memory the size of one page or larger. The guard page causes a segmentation fault upon any access. As a result, any attempt by an attacker to overwrite adjacent memory in the course of exploiting a buffer overflow causes the vulnerable program to terminate rather than continue execution of the attacker-supplied code. Guard pages are implemented by a number of systems and tools, including OpenBSD, Electric Fence, and Application Verifier (each of which is discussed further in this content area).

Guard pages have a high degree of overhead because they fragment the kernel's memory map and can increase the amount of virtual space considerably. Their effectiveness depends on the size and pattern of allocations; they are often more effective as a debugging facility than an operational security measure.

References

[ISO/IEC 99] ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming Languages — C.* International Organization for Standardization, 1999.

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/268-BSI.html (Plakosh, Daniel)

Pearson Education, Inc. Copyright